PATENT ABSTRACTS OF JAPAN

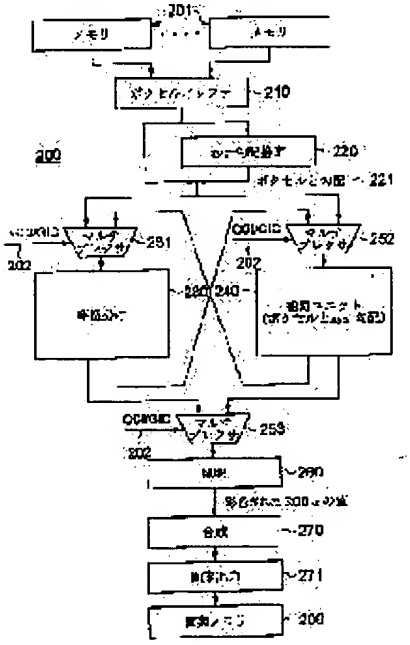
(11)Publication number : **2002-183747**
(43)Date of publication of application : **28.06.2002**

(51)Int.CI. G06T 15/00

(21)Application number : **2000-365880**
(22)Date of filing : **30.11.2000**

(71)Applicant : **TERARECON INC**
(72)Inventor : **LAUER HUGH C**
SEILER LARRY D
KNITTEL JAMES M
CORRELL KENNETH W
GASPARAKIS HARRY
SIMHA VIKRAM
BHATIA VISHAL C

(54) VOLUME RENDERING PIPELINE



(57)Abstract:
PROBLEM TO BE SOLVED: To realize a rendering pipeline for volume rendering allowing user's selection of processing order in stages of the pipeline.
SOLUTION: This volume rendering pipeline includes the plural processing stages such as gradient estimation 220, interpolation 240, classification 230, lighting 260 and composition 270. A first multiplexer 251 connects an output part of the first stage 240 to an input part of the second stage 230. A second multiplexer 252 connects an output part of the second stage 230 to an input part of the first stage 240. An input part of a third inultiplexer 253 is connected to the output part of the first stage 240 and the output part of the second stage 230. The multiplexers 251, 252, 253 respond to a selection signal and set the rendering pipeline to process volume data set.

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号
特開2002－183747
（P2002－183747A）
(43)公開日　平成14年６月28日(2002.6.28)

(51)Int.Cl.⁷	識別記号	ＦＩ		テーマコード（参考）
Ｇ０６Ｔ　15/00	２００	Ｇ０６Ｔ　15/00	２００	５Ｂ０８０

審査請求　未請求　請求項の数13　ＯＬ　外国語出願　（全 41 頁）

(21)出願番号　特願2000－365880(P2000－365880)

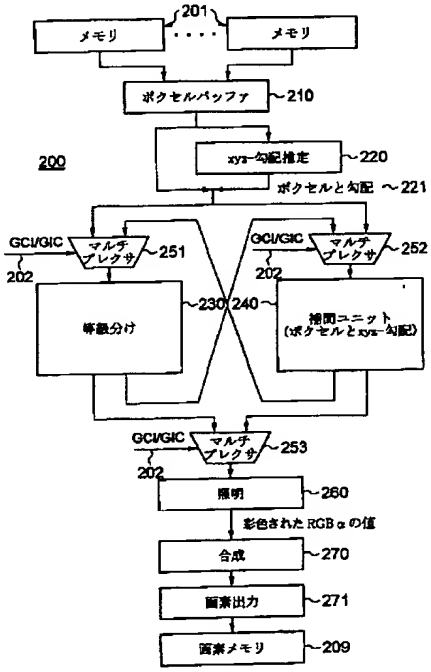
(22)出願日　平成12年11月30日(2000.11.30)

(71)出願人　501184364
テラレコン・インコーポレイテッド
ＴｅｒａＲｅｃｏｎ，　Ｉｎｃ．
アメリカ合衆国、カリフォルニア州、サン・マテオ、キャンパス・ドライブ
2955、スイート　325
(72)発明者　ヒュー・シー・ラウアー
アメリカ合衆国、マサチューセッツ州、コンコード、ボーダー・ロード　69
(74)代理人　100057874
弁理士　曾我　道照　（外７名）

最終頁に続く

(54)【発明の名称】　ボリュームレンダリングパイプライン

(57)【要約】　（修正有）
【課題】ボリュームレンダリングに関し、特にパイプラインのステージにおける処理順番がユーザによって選択可能になっているレンダリングパイプラインを実現する。
【解決手段】　ボリュームレンダリングパイプラインは、勾配推定２２０、補間２４０、等級分け２３０、照明２６０及び合成２７０等の複数の処理ステージを含んでいる。第１マルチプレクサ２５１は、第１ステージ２４０の出力部を第２ステージ２３０の入力部に接続している。第２マルチプレクサ２５２は、第２ステージ２３０の出力部を第１ステージ２４０の入力部に接続している。第３マルチプレクサ２５３は、入力部を第１ステージ２４０の出力部と第２ステージ２３０の出力部に接続しており、マルチプレクサ２５１，２５２，２５３は、選択信号に応答して、ボリュームデータセットを処理するためにレンダリングパイプラインを設定する。



【特許請求の範囲】
【請求項１】　ボリュームデータセットをレンダリングするための複数のステージを有したボリュームレンダリングパイプラインであって、
第１ステージの出力部を第２ステージの入力部に接続している第１マルチプレクサと、
上記第２ステージの出力部を上記第１ステージの入力部に接続している第２マルチプレクサと、
入力部を上記第１ステージの出力部及び上記第２ステージの出力部に接続した第３マルチプレクサととを備え、
上記第１、第２及び第３のマルチプレクサが、上記第１及び第２のステージを介して上記ボリュームデータセットの処理順番を選択する選択信号に応答することを特徴とするボリュームレンダリングパイプライン。
【請求項２】　上記複数のステージは、勾配推定ステージと、補間ステージと、等級分けステージと、照明ステージと、合成ステージとを有していることを特徴とする請求項１に記載のボリュームレンダリングパイプライン。
【請求項３】　上記選択信号は、上記等級分けステージの前に上記補間ステージを接続することを特徴とする請求項２に記載のボリュームレンダリングパイプライン。
【請求項４】　上記選択信号は、上記補間ステージの前に上記等級分けステージを接続することを特徴とする請求項２に記載のボリュームレンダリングパイプライン。
【請求項５】　勾配バッファを使用している勾配補間子は、ＲＢＧαバッファを使用しているＲＧＢα補間子と並列状態で作動することを特徴とする請求項１に記載のボリュームレンダリングパイプライン。
【請求項６】　上記第１、第２及び第３のマルチプレクサは、上記パイプラインの特定ステージをバイパスすることを特徴とする請求項１に記載のボリュームレンダリングパイプライン。
【請求項７】　更に、上記パイプラインの第１ステージと最終ステージに接続されたメモリを備え、該メモリが、上記パイプラインによる処理の前後に上記ボリュームデータセットを記憶することを特徴とする請求項１に記載のボリュームレンダリングパイプライン。
【請求項８】　上記メモリは、レンダリングされた画像を記憶することを特徴とする請求項１に記載のボリュームレンダリングパイプライン。
【請求項９】　上記ボリュームのデータセットは、複数のボクセルを有しており、各ボクセルは、複数のフィールドを有していることを特徴とする請求項１に記載のボリュームレンダリングパイプライン。
【請求項１０】　各フィールドは、上記ボクセルに、関連するずれと幅とを有していることを特徴とする請求項９に記載のボリュームレンダリングパイプライン。
【請求項１１】　特定のフィールドは、ボリュームカテゴリを記憶することを特徴とする請求項９に記載のボリュームレンダリングパイプライン。
【請求項１２】　各フィールドは、各々異なり関連した補間関数に従って補間されることを特徴とする請求項９に記載のボリュームレンダリングパイプライン。
【請求項１３】　上記勾配推定ステージは、複数のフィールドを有した特定のボクセルから勾配成分を抽出することを特徴とする請求項２に記載のボリュームレンダリングパイプライン。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】この発明は、一般にボリュームのレンダリングに関し、特にパイプラインのステージにおける処理順番がユーザによって選択可能になっているレンダリングパイプラインに関する。
【０００２】
【従来の技術】ボリュームレンダリングの紹介ボリュームレンダリングは、３次元データが目視化される必要があるコンピュータグラフィックスの用途にしばしば使用される。ボリュームデータは、身体対象物や医療対象物についての、又は環境上の、地球物理学上の、又は他の科学上のモデルについての走査による画像データであり、そこでは、データを目視化することでデータによって表される潜在した実世界構造を理解しやすくするものである。
【０００３】ボリュームレンダリングによって、身体対象物やモデルの内部構造と、外表面の特徴は目視化される。ボクセルは、通常、ボリュームレンダリングで使用される基礎的なデータ細目である。ボクセルは、対象物やモデルの特定の３次元部分を表すデータ細目である。各ボクセルの座標（ｘ、ｙ、ｚ）は、表される対象物やモデルの内部における諸々の位置に対してボクセルを写像する。
【０００４】ボクセルは、対象物又はモデルの或る特定の強度値を表している。或る従来技術に係るボリュームに対して、強度値は、密度、組織のタイプ、弾性又は速度等の幾つかの異なったパラメータの内の特定対象とすることができる。レンダリング中に、ボクセルの値は、見るために２次元画像面上に投影されることになる色と不透明度（ＲＧＢα）の値に強度値に従って変換される。
【０００５】レンダリング中にしばしば使用される一つの技法は、レイキヤスチング（光線投射）である。一組の仮想光線がボクセルのアレーを通して投光される。光線は、見る人の目又は仮想面から発生するものとしている。ボクセル値は、光線に沿った諸々の点に対して繰り返しサンプリングされ、またサンプリングされた値を画素値に変換する色々な技法が知られている。代わりに、ボクセル値は、直接ＲＧＢαのボクセルに変換され、それらが、次に光線に沿って繰り返しサンプリングされ且つ画素値に蓄積されるものがある。いずれの場合も、ボ

リュームデータの処理は、後から前へ、又は前から後に進行できる。

【0006】レンダリングパイプラインボリュームレンダリングは、ソフトウェア又はハードウェアによって行われる。一つのハードウェアの実行では、ハードウェアは多ステージパイプラインとして構成されており、1998年11月12日にカプラー氏等によって出願された米国特許出願第09/190,643号の『実時間のボリュームレンダリングシステムにおける中間値の急速記憶と検索』を見て下さい。

【0007】図1は、ボクセル値がボクセルメモリ101に記憶されているパイプライン100を図示している。ボクセル値は、先ずスライスとしてパイプラインのボクセルバッファ110内に読み込まれる。勾配のz成分は、異なったスライスのボクセル間における中央差を取ることでステージ115で推定される。次に、ボクセル値とz勾配の両方共、光線に沿ったサンプリング点でこれらの値を算定する補間ステージ120に移される。次に、勾配のx成分及びy成分は、ステージ130において補間されたサンプル値から算定される。これらは、サンプル値及び補間されたz勾配と一緒に等級分けステージ140に移され、次に色調合わせステージ145に移され、そこで照明処理が行われて彩色されたサンプルを表すRGBα値を発生する。最後に、彩色されたサンプルは、合成ステージ150において光線に沿って結合され、画素メモリ109に記憶された基礎面のための画素値を発生する。

【0008】

【発明が解決しようとする課題】そのようなパイプラインは、データ処理順が色々なステージの配列によって固定されるので悪影響を受ける。更に、ただ補間されたサンプルのみが等級分けされるようにボクセル値は補間される。異なった走査の特徴的属性から得られる多くのボリュームを同時にレンダリングすることは不可能である。更に、ボクセルデータのフォーマットは固定されている。勾配のフィールドは、固定されたフォーマットのボクセルデータから得られる。この従来技術のパイプラインについて改善することが望まれている。

【0009】

【課題を解決するための手段】本発明は、複数の処理ステージを有したボリュームレンダリングパイプラインを提供する。ステージには、勾配推定ステージと、補間ステージと、等級分けステージと、照明ステージと、合成ステージとが含まれる。それらステージは、マルチプレクサによって互いに接続されている。第1マルチプレクサは、特定ステージの出力部をもう一つ別のステージの入力部に接続し、また第2マルチプレクサは、特定ステージの入力部をもう一つ別のステージの出力部に接続している。それらマルチプレクサは、選択的にパイプラインのステージを所定の順番で接続し、ボリュームデータセットを処理するためにレンダリングパイプラインを設定する。

【0010】本発明の一つの局面では、ボリュームデータセットのボクセルは、等級分けされる前に補間され、またもう一つ別の局面では、ボクセルは、それらが等級分けされる前に補間される。

【0011】

【発明の実施の形態】図2は、本発明に係る設定可能なレンダリングパイプライン200のトップレベルのブロック線図を示している。パイプライン200は、ボクセルメモリ201から入力としてサンプル又はボクセルを採取し、画素メモリ209に出力として画素を記憶する。ボクセル又はサンプルは、同時にボリュームスライスとしてボクセルバッファ210に読み込まれる。勾配は、ステージ220においてxyzの成分を求めるべく推定される。ここでは、従来技術とは対照的に、全ての勾配成分は、ボクセル又はサンプルの値に基づいて推定されるが、補間されたサンプルに基づいては推定されないようになっている。

【0012】ボクセル221は、第2ステージ230で等級分けされる。ボクセルと勾配の補間は、第1ステージ240において行われる。勾配の推定と補間は、一次線形オペレーションとなっていて、従って互換される。本発明の効果として、用途によってボクセルと勾配は色々なステージによって処理される順番が決められる。2つの異なった処理順番に対する理由は、次のようになっている。

【0013】スキャナーは、一般にスペースの小さな領域に渡って積分を表す物理学的計測値を作ることで、サンプリングされたデータを獲得する。隣接したボクセルは、しばしばオーバラップした領域の積分を表している。これらの場合、先ずボクセルを補間し、次にその結果を等級分けするのがより高精度となる。具体的には、所謂2つの異なった組織の境界におけるボクセルは、各組織の物性の平均を表すようである。補間する際に実施される事は、平均点を移動することだけである。これらを役立てるために、等級分け機能は、2つの組織のタイプに割り当てられた色と不透明度との間から連続した推移を提供すべきである。

【0014】時々、サンプリングされたデータは、例えば骨とか筋肉とか軟骨等の別個の物体に予め区分（手で又は自動的に）されることがある。もし、そのようなデータが与えられると、2つの別々に識別された組織の間で補間を行って、そこに実際には存在しない第3の組織を得ることになるのは不適当な事であろう。これらの場合、先ずボクセルを等級分けし、次に結果的に生じる色を補間するのがより優れている。これは、隣接した組織のタイプの物性の平均という程の事には成らないが、むしろそれらの境界で色が混り合う程度の事である。

【0015】従って、パイプライン200は、第1ステ

ージ２４０からの出力を第２ステージ２３０の入力部に接続しているマルチプレクサ２５１を有している。同様に、マルチプレクサ２５２は、第２ステージ２３０の出力部を第１ステージ２４０の入力部に接続している。マルチプレクサ２５３は、第２ステージ２３０又は第１ステージ２４０のいずれかからの出力を選択する。

【００１６】ステージの接続の順番は、３つのマルチプレクサ２５１～２５３への選択信号（ＧＣＩ／ＧＩＣ）２０２に左右される。いずれの場合も、補間された勾配と補間されたサンプルは、照明ステージ２６０に移される。彩色されたＲＧＢαの値は、ステージ２７０で合成され、また出力画素２７１は、画素メモリ２０９に記憶される。

【００１７】パイプライン２００では、勾配推定ステージ２２０の出力は、勾配の３の成分となっている。これらは、一組のマルチプレクサを介して等級分けステージと補間ステージの両方に移される。これらのステージの各々の出力は、各々他方のステージの入力部に、更にもう一つ別のマルチプレクサを介して照明ステージ２６０にも移される。

【００１８】３つのマルチプレクサ２５１～２５３を適切に選択することで、補間の前に等級分けを、又は等級分けの前に補間を、又は補間のみを、又は等級分けのみを行うことができる。さて、これらの異なった作動モードをより詳細に説明する。

【００１９】ＧＩＣ－勾配、補間、等級分け図３は、ＧＩＣモード用のレンダリングプロセスを図示している。このモードは、信号２０２によって選択される。ＧＩＣモードでは、勾配が先ず推定され、次いで補間が、勾配とボクセルについて平行して行われ、その後にボクセルの等級分けが行われる。

【００２０】レンダリングは、一度に、即ち３２×３２の光線グループでの切断作業に進む。光線の切断作業を行うために、ボクセルのスライスが、同時に２つのスライスづつボクセルバッファ２１０に読み込まれる。各スライスは、その切断に必要とされるボクセルのみを有している。スライス当りのボクセルの数は、視線方向とサンプリングの繰り返し頻度によって変わる。必要とされるボクセルの最大数は、３７×３７のボクセルである。

【００２１】ボクセルをボクセルバッファ２１０に読み込むには、指定スライスの全帯域を使いきることになる。従って、いずれの所定時にも、一対のスライスが新しいボクセルを受け入れるのに専用化されており、他方、他の２対のスライスが、パイプラインの次のステージ、即ち勾配推定ステージ２２０とボクセル補間ステージ２４０に供給するのに専用化されている。

【００２２】図３において、ｓ＋１とｓ＋２のラベルの付いたスライスは、新しいボクセルを受け取っており、他方スライスｓ、ｓ－１及びｓ－２は、ボクセルを後続ステージに提供している。スライスｓ－３は、前のスライス中に使用されていたものであり、現在空になっている。スライスｓ－２の作業が完了すると、スライスｓ－２も空としてマークされる。次に、スライスｓ－２、ｓ－３がボクセル読み込みの新しい指定地になり、他方処理モジュールがスライスｓ＋１、ｓ、ｓ－１からそれらの入力の採取を開始する。

【００２３】勾配推定ステージ２２０は、スライスｓ－１におけるボクセル点における勾配をスライスｓ、ｓ－２におけるボクセル値から推定する。これらの勾配は、次にユニット２２８による補間のために、２つのスライスの勾配バッファ２２５に記憶される。スライスｓ－２におけるボクセル点の勾配は、前のスライス相互作用中に勾配バッファに記憶されている。

【００２４】次に、ボクセルの補間２４０と勾配補間２２８が同時に進行する。即ち、サンプルのスライスが、ボクセルのスライスｓ－１、ｓ－２からのボクセル値を補間することで得られる。サンプルの現用のスライスにおけるサンプリング点における勾配は、勾配バッファにおいて勾配スライスｓ－１と勾配スライスｓ－２の勾配を補間することで得られる。サンプルと補間された勾配の一つより多いスライスが、同じ対のボクセルと勾配のスライスから得られる。

【００２５】補間されたボクセルの値は、次に等級分けステージ２３０に適用され、その等級分けステージ２３０は、それらの値を補間されたＲＧＢαの色値に変換する。ＲＧＢα値と勾配は、次に照明ステージ２６０に供給され、最後に合成ステージ２７０に供給される。

【００２６】オプションモードでは、いずれかのボクセルのフィールドから勾配を推定するのに勾配推定ステージ２２０を使用する代わりに、勾配は、例えば３つのボクセルフィールドから直接的に抽出される。

【００２７】ＧＣＩ－勾配、等級分け、補間

図４は、更に信号２０２によって選択されるＧＣＩモード用のレンダリングアルゴリズムを図示している。ボクセルバッファ２１０は、上述のように充足され、また勾配推定プロセスも同じである。相違点は、ボクセルの等級分けと補間の側に存在している。特に、生のボクセルは、ボクセルスライスバッファから、即ちスライスｓ－１から採取され、即座に等級分けステージ２３０によってＲＧＢα値に変換される。これらは、次に２つのスライスのＲＧＢαバッファ２３５に記憶される。ＲＧＢαバッファ２３５は、２つのスライスの勾配バッファ２２５に対して並列になっている点に注目して下さい。

【００２８】２つのスライスが、次にＲＧＢαの補間ステージ２４０に入力され、そこで補間されたＲＧＢα値は、勾配の補間と平行して発生される。これらは、次に補間された勾配と共に、照明ステージと合成ステージとに適用されて、基礎面の画像の画素出力２７１を発生する。これらの最後の２つのモジュールは、図２～３のものと同じである。

【００２９】一つの付加的なオプションモードも可能である。このオプションモードでは、多くの、例えば４つのボクセルのフィールドが、生のＲＧＢα値として補間されている。この場合、勾配は、アルファ（α）のフィールドから推定され得よう。幾つかのボリュームデータセットとレンダリングモードに対しては、例えば等級分け後にパイプラインの後ステージにおいて勾配を推定するのが最良であることに注目すべきである。

【００３０】マルチプレクサ２５１～２５３は、他のステージのために複製され得ることは明白であるべきである。ステージは、再設定可能なレンダリングパイプラインを提供するために幾つかの異なった順番で選択され、接続される。幾つかのステージは、ボリュームデータを全く処理しないために、例えば、幾つかのレンダリングのために、勾配推定と照明とがスキップされるように、選択から外される。

【００３１】フレキシブルなボクセルフォーマット

図５に示されているように、設定可能なレンダリングパイプライン２００によって使用されるようなボクセル５００は、複数のフィールド（V_1、・・・、V_n）５０１～５０９を有している。フィールド５０１～５０９の各々は、ボクセル５００においてずれと幅として特定される。いずれのフィールドも、フィールドV_1、V_2に対して図示されているようにオーバラップしている。フィールドは、いずれの順番でも列挙されるようになっている。

【００３２】フィールドは、表される３次元の対象物又はモデルの異なった属性を説明する。例えば、もし対象物が人の頭の場合、その時、フィールド５０１～５０９は、各々、ＣＴ、ＭＲＩ、ＰＥＴ、ＳＰＥＣＴ及び超音波スキャンから獲得される強度値を記憶するものであり、即ち、各ボクセルは、単一のボリューム表示において５つの異なった走査によるデータの強度値を記憶できる。代わりに、走査によるデータは、各ボクセルが一つのフィールドに寄与している多くの別々のボリュームとして記憶される。

【００３３】幾つかのフィールドは、ボリュームが、例えば手動での、又は半自動的な、又は自動的な区分として区画される方法に関連したカテゴリフィールドとなっている。医療の用途では、区分は、骨や組織等を類別している。科学的用途では、区分は、特定の方法でレンダリングされる副組立体や他の部分を同定するものである。科学的モードのために、フィールドは、例えば圧力、速度、角モーメンタム、弾性、密度、温度及び粘度等の、特定の目視化に使用される状態変数を記憶する。いずれのボリュームデータセットのためにも、フィールドは、更にＲＧＢα値や、深度や、３Ｄステンシル、陰、霧、ボクセル化され且つ填め込まれた幾何学的なボリュームや、勾配も記憶するものである。

【００３４】我々の発明に係るマルチ－フィールドのボクセル５００に対して、ユーザは、勾配算定にどのフィールドを使うかを特定できる。勾配の各成分に対して、我々は、その成分を求めるためにどのボクセルフィールドを使うかを我々は特定できる。

【００３５】マルチ－フィールドの目視化のために、通常ボクセル内でフィールドは別々に補間するのが望ましい。更に、ボクセル内の各フィールドは、例えば強度フィールド用の３つの線形やカテゴリフィールド用に最も近い隣接補間法等の異なった補間機能を適用している。ここで説明されているようなフレキシブルなボクセルは、一様な方式でボクセルフォーマットの全ての特別ケースを扱うために共通のフレームワークを使用可能としている。

【００３６】フィールドフォーマットのレジスタ

図６は、本発明に係る多数のフィールドのボクセルを使用可能にするフィールドフォーマットのレジスタ６００を示している。一実施例では、ボクセルのフィールドは、フィールドフォーマットレジスタの説明によって定義される。これは、フィールド（４－ビットのニブルの）６１０のサイズと、そのボクセル６２０内部におけるフィールドの位置と、画素（更に、４－ビットのニブルの）と、フィールドがその意図された使用のために特定されたものとは異なったサイズとなっている時にすべきもの（制御６３０）とを定義する８－ビット（７：０）の記述子となっている。

【００３７】コントロールビットは、フィールドが通って行くパイプラインのデータ通路に適合するためにフィールドがどのように適応されるかを定義している。フィールドは、分数の計算を繰り返すことで、又は最高有効分数又は最低有効分数のいずれかからビットを加減することで変更される。

【００３８】コントロール＝０：　生のボクセルのフィールドは、範囲［０.. １］における数を表している符号無しの繰り返し分数として扱われる。データ通路に適合するために繰り返し分数を展開したり、又は縮小するために、繰り返し分数の計算が、概算に適用されて、それによって数を前進のより少ない又はより多いビットで表す。繰り返し分数の数表示については、以下により詳細に説明する。

コントロール＝１：　ボクセルのフィールドは、範囲［－１...＋１］における符号付き繰り返し分数として扱われる。

コントロール＝２：　生のボクセルのフィールドは、データ通路に適合するためにその最低有効ビット（ＬＳＢ）で展開されるか、又は切り捨てられる。最高有効ビットは、保存される。

コントロール＝３：　生のボクセルのフィールドは、データ通路に適合するためにその最高有効ビット（ＭＳＢ）で展開されるか、又は切り捨てられる。最低有効ビットは、保存される。

【００３９】繰り返し分数の数表示
多くのグラフィックスの用途は、色や、透明度や、又は０から１までの範囲の値を有している他のパラメータを表すために、一定の幅の２進法の数を使用している。
【００４０】Ｒを２進法の数におけるビット数とし、またＶをこれらビットに記憶された符号無しの２進値とする。それで、Ｆ＝Ｖ／（2^R － １）は、範囲［０．．１］における有理数と成っている。即ち、Ｖが０の時、Ｆは０であり、またＶがその最大可能値の（2^R － １）の時、Ｆは１である。この表示は、従来技術では公知である。例えば、OpenGL（公開グリーンランド）明細書は、不動点表示の特別種類としてそれを参照している。
【００４１】ここで説明された表示を通常の不動点表示から明確に識別するために、用語の『繰り返し分数』が使用される。その名の用語は、Ｆを不動点の２進法の分数に展開すると言うことが、０．ＶＶＶＶＶＶ．．．を、即ち、Ｒ－ビット値のＶを２進法の小数点の右に無限に繰り返す２進法の分数を発生することであると言う事実から由来する。
【００４２】繰り返しの分数は、Ｒより多くのビットで表され、また符号も付けられる。その場合、Ｒが暗に意味されたスケール因数（2^R － １）を定義しているので、Ｒは、数の『繰り返し精度』である。これで、Ｆが範囲［０．．１］以外の値を持てるようにしている。この場合、２進法の不動点表示は、整数値と、その後に続くＲ－ビットの無限に繰り返す２進法値とから構成されている。同じ精度を有した繰り返し分数は、通常の整数と同じ方法で加減される。
【００４３】繰り返し精度の変更を含んだ他の計算作業は、先ず各々の繰り返し分数のためにＦを演算し、通常の計算を実施し、次いでその結果生じる値に（2^R － １）を掛けて結果の繰り返し精度Ｒを求めることで実施され得る。より効果的なフォームが、繰り返し分数についての作業のために存在している。例えば、Ｒから２Ｒに繰り返し精度を２倍するには、単にＶ＋（Ｖ＜＜Ｒ）を演算するだけでよい。
【００４４】ボクセルフォーマット
本発明に係るパイプラインは、ボクセルの入力フォーマットについて広い範囲を許容している。入力のボクセルは、８、１６、３２又はより大きなビット量とできる。フィールドは、幅において４、８、１２又は１６ビットとでき、またボクセル内で４－ビット境界上で整合されている。全てのボクセルフィールドは、一般に１２ビットの広がりとなっているそれらのデータ通路を適合させるために調整される。
【００４５】ボクセルのフォーマットについて、ボクセルフォーマットのレジスタ７００において説明する。３２－ビットのボクセル８００のフォーマットの例は、図８に図示されている。図７における例のボクセルのためのフォーマットレジスタでは、フィールド０はビットの１１：０を占め、フィールド１はビットの２３：２０を占め、フィールド２はビットの３１：２４を占め、またフィールド３はフィールド１とオーバラップしていて、ビット１５：８を占めている。図８に示されている例に対して、フィールドフォーマットレジスタにおけるフィールドについての説明は、次のようになっている：
フィールド３：サイズ＝１　位置＝２
フィールド２：サイズ＝１　位置＝６
フィールド１：サイズ＝０　位置＝５
フィールド０：サイズ＝２　位置＝０
【００４６】長所
フレキシブルなフォーマットボクセルを有する事と再設定可能なパイプラインとは、幾つかの長所を有している。例えば、ボクセルにおけるフィールドの一つは、カテゴリのビットのために使用される。これらは、或る特定の組織の一部分として、副組立体として、又はボリュームの他の部分としてボクセルを識別するボクセルでの余分なビットである。それらは、補間されないが、しかし、それらは、ボクセルに対するＲＧＢαの割当てに大いに寄与する。カテゴリのビットが使用される時、等級分けは、通常は補間に先んじて行われる。
【００４７】フレキシブルなボクセルのフォーマットは、勾配がボクセルのいずれかの選択されたフィールドから推定されるようにしている。これらの場合、中央差等の回旋カーネルは、各ボクセルの選択されたフィールドに適用され、そのボクセルの勾配のｘ、ｙ及びｚの成分を得る。これらの勾配は、次に、サンプリング点における勾配を得るために生の又は等級分けされたボクセルと平行に補間される。これらは、次にＲＧＢα値と一緒に照明ステージ２６０に適用され、最後に合成ステージ２７０に適用される。
【００４８】幾つかの応用で、また幾つかのボリュームに対して、移動中におけるボクセルのフィールドから勾配を推定する代わりに或る他の方式で決定される勾配を使用する方がより良いと思われる。フレキシブルなフォーマットのボクセルは、このモードに配慮している。特に、勾配は、高レベルの精度を持ったボクセルのために予め演算される。これらの予め演算された勾配は、ボクセルのフィールドの一つに記憶される。これらの予め演算された勾配は、推定ステージをバイパスするが、しかし丁度移動中において推定される勾配として照明ステージにおいて補間され、適用される。
【００４９】更に別の変形例では、予め演算された勾配は、勾配推定のために回旋カーネルに適用される。これは、ボリュームデータセットの第２の部分的な派生物を採用すると言う効果を有する。同じ技法によって、より高い順番の部分的な派生物が得られる。そのような派生物は、ボリュームデータセットにおける弱い面を抽出するに役立つ。以下、図９に対して説明するように、これは、『マルチーパス』操作において（１～３の）多数回

に渡ってパイプラインにボリュームデータセットを通すことで実施される。各パスは、異なった組のレンダリングパラメータでボリュームデータセットを処理する。

【００５０】フレキシブルなフォーマットのボクセルを処理するパイプライン２００は、更に、『ＲＧＢα』のボクセルとして予め等級分けされ、与えられたボリュームデータセットを入れる。２つの代替案が役立つ。一方では、勾配推定と等級分けと色調合わせは、全体的にスキップされ、またボクセルは、ただ単に補間されるだけである。他方の代替案では、勾配は、α値又は輝き関数から推定され、またＲＧＢα値の色調合わせは、照明関数に従って実施される。本発明に係るパイプライン２００も、ボクセル－勾配値とＲＧＢα値の両方を補間することができる点に注目してください。

【００５１】マルチ－パスのボリュームレンダリング

レンダリングパイプライン２００は、更に、パイプライン２００のステージの幾つかを、しかし必ずしも全てでは無く、通過してボクセルメモリに戻ったボリュームデータセットを書き出すこともできるもので、図９を見て下さい。例の使用では、設定されたパイプラインは、一組のボクセルをＲＧＢα値にレンダリングする。光線に沿って合成される代わりに、ＲＧＢαは、３次元の画素アレーのＲＧＢαのボクセルとしてメモリ２０１に戻して記憶される。次に、ボリュームは、異なった組のレンダリングパラメータでレンダリングされる。このプロセスは、最終のボリュームがメモリ２０１に存在するまで繰り返す。最終の蓄積された値は、次に最終回でレンダリングされて画素メモリ２０９に画像を発生させる。

【００５２】この技法は、急速再サンプリングなどの幾つかの特長を実施できるようにしている。急速再サンプリングによって、ボリュームデータセットは、ホストプロセッサーとソフトウエアに依存する代わりに、ボリュームレンダリングパイプライン及びボリュームメモリの速度とパワーを使用して異なった解像度に対して再サンプリングされる。

【００５３】マルチ－パスレンダリングは、ボリュームに関する複雑な陰影を発生するために使用される。各光源に一つのパスが必要とされる。各パスは、出力のボリュームデータセットに蓄積され、次いで最終パスは、結果を補間し、それらを基礎面上に投影する。マルチ－パスレンダリングでは、現在のパスからのボリュームデータセットの出力は、メモリに既に存在しているボリュームデータセットと結合される。これは、時々、単なる書き出し操作よりはむしろ読み取り－変更－書き出し操作を必要とする。

【００５４】マルチ－チャンネルレンダリング

パイプライン２００は、更にマルチ－チャンネルのデータセットも処理する。超音波用途や人工地震用途などの幾つかの走査技法は、各々それ自身の等級分けを有した一種類以上のデータを持っている。レンダリング中に、これらのデータセットは、互いに重ね合わされ、ボクセル毎に結合されることになる。

【００５５】より具体的には、合成ステージ２７０は、２つのモデルのいずれかで作動できる。第１モードでは、ＲＧＢα値は、以前に記憶された画素値と結合され、また第２モードでは、ＲＧＢα値の光線は、蓄積され、またその結果は、或る以前のパスからの以前に記憶された画素値と結合される。

【００５６】パイプラインの概略構造

一般に、パイプライン２００とメモリ２０１との関係は、図９に示されているように成っている。ここでは、パイプライン２００のステージ（ステージ$_0$、ステージ$_1$、．．．、ステージ$_n$）１～３は、特定のレンダリング用途に対して、ステージが選択信号５によって順番付けされるようにマルチプレクサ４によって互いに接続される。パイプラインへの入力は、メモリ６に記憶された生のボリュームデータセットである。パイプラインの出力は、改造されたボリュームデータセットである。パイプラインとメモリの間を通ったボリュームデータセットの個々のデータ細目は、フレキシブルなフォーマットのボクセル７となっている。ボリュームデータセットは、マルチプルパスによって処理される。

【００５７】色々な他の適合化及び改造が、本発明の精神と技術的範囲内で実施される。従って、本発明の真正な精神と技術的範囲内に入るような全ての変形と改造を保護することは従属項の目的である。

【図面の簡単な説明】

【図１】　従来技術のレンダリングパイプラインのブロック線図である。

【図２】　設定可能なレンダリングパイプラインのブロック線図である。

【図３】　等級分けの前に補間を行うパイプラインである。

【図４】　補間の前に等級分けを行うパイプラインである。

【図５】　フレキシブルなフォーマットのボクセルのブロック線図である。

【図６】　フィールドのフォーマットレジスタのブロック線図である。

【図７】　ボクセルのフォーマットレジスタのブロック線図である。

【図８】　例のフォーマット化されたボクセルである。

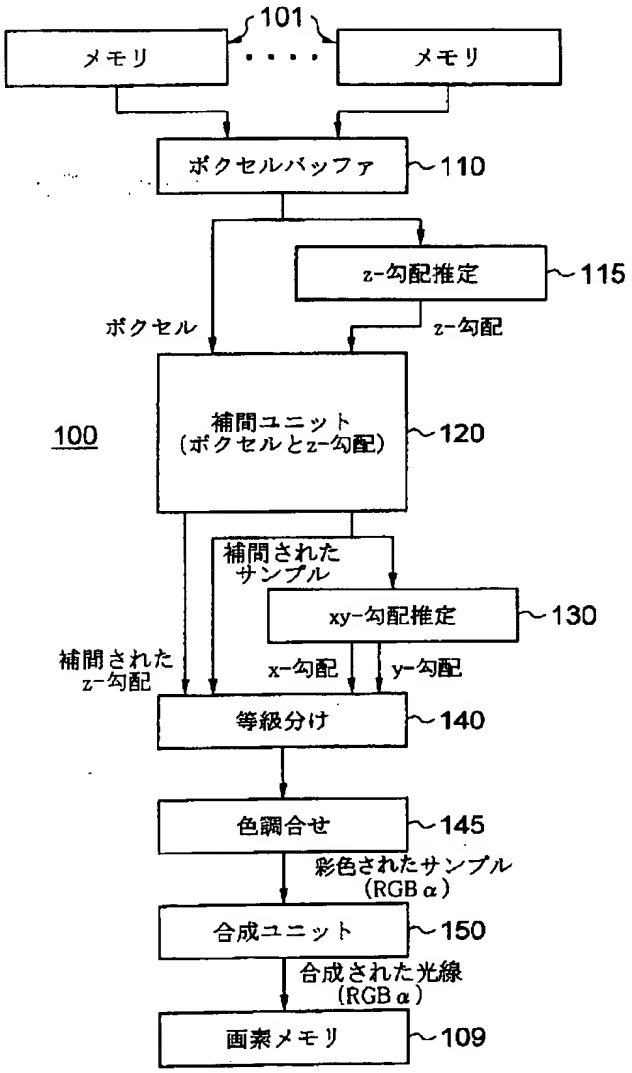
【図９】　ボリュームメモリに接続された設定可能なレンダリングパイプラインのブロック線図である。

【符号の説明】

１～３　ステージ、５　選択信号、６　ボリュームデータセットメモリ、２００　ボリュームレンダリングパイプライン、２１０　ボクセルバッファ、２２５勾配バッファ、２２０　勾配推定ステージ、２２８　勾配補間子、２３０　等級分けステージ、２３５　ＲＧＢαバッ

ファ、２４０　補間ステージ（ＲＧＢα補間子）、２５１　第１マルチプレクサ、２５２　第２マルチプレクサ、２５３第３マルチプレクサ、２６０　照明ステージ、２７０　合成ステージ、５００ボクセル、５０１～５０９　フィールド。

【図１】

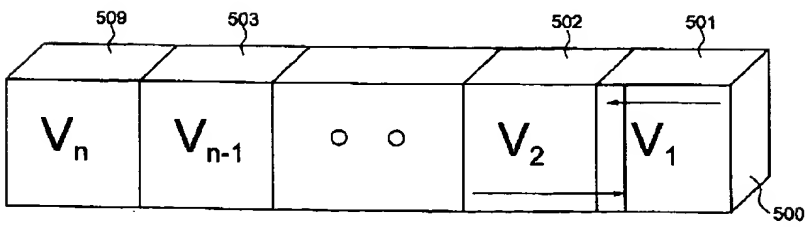


【図７】

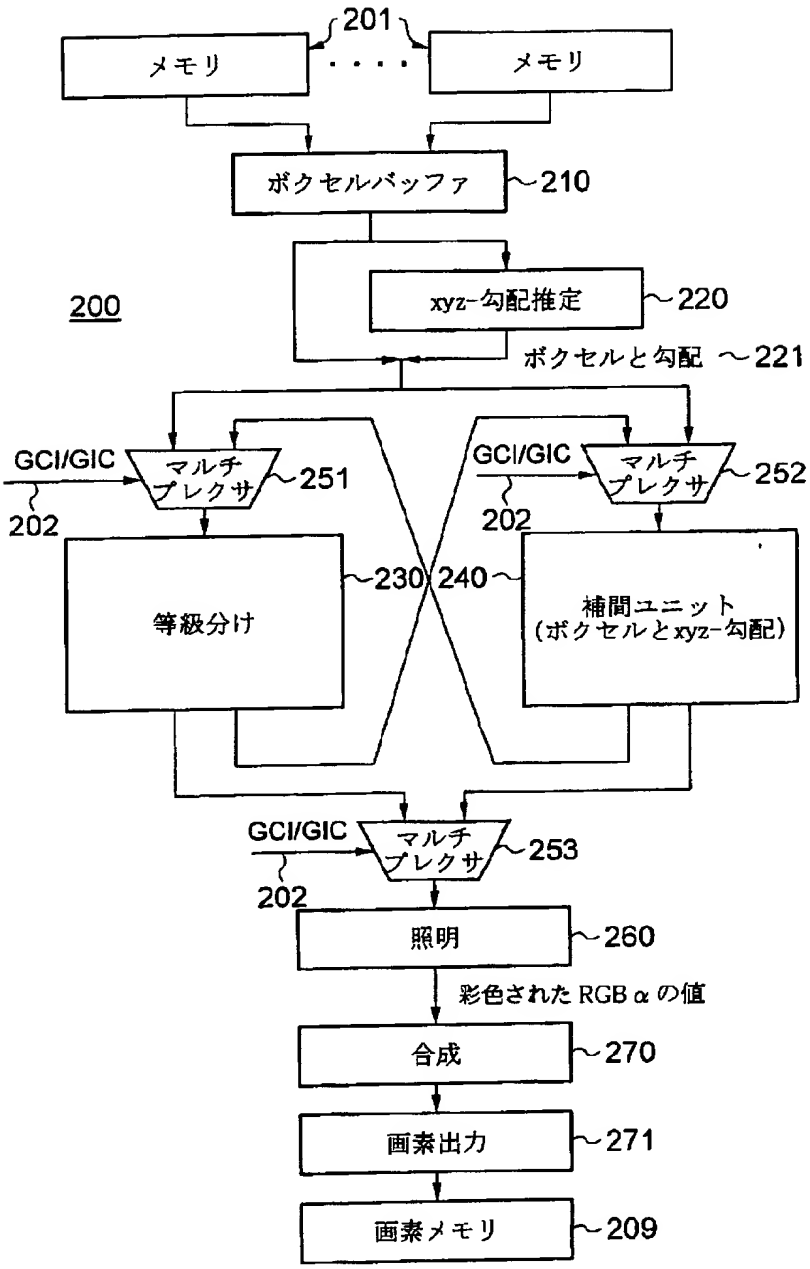
700

ボクセルフォーマットのレジスタ	
フィールドの名称	説明
フィールド4	フィールド4の記述子
フィールド3	フィールド3の記述子
フィールド2	フィールド2の記述子
フィールド1	フィールド1の記述子
フィールド0	フィールド0の記述子

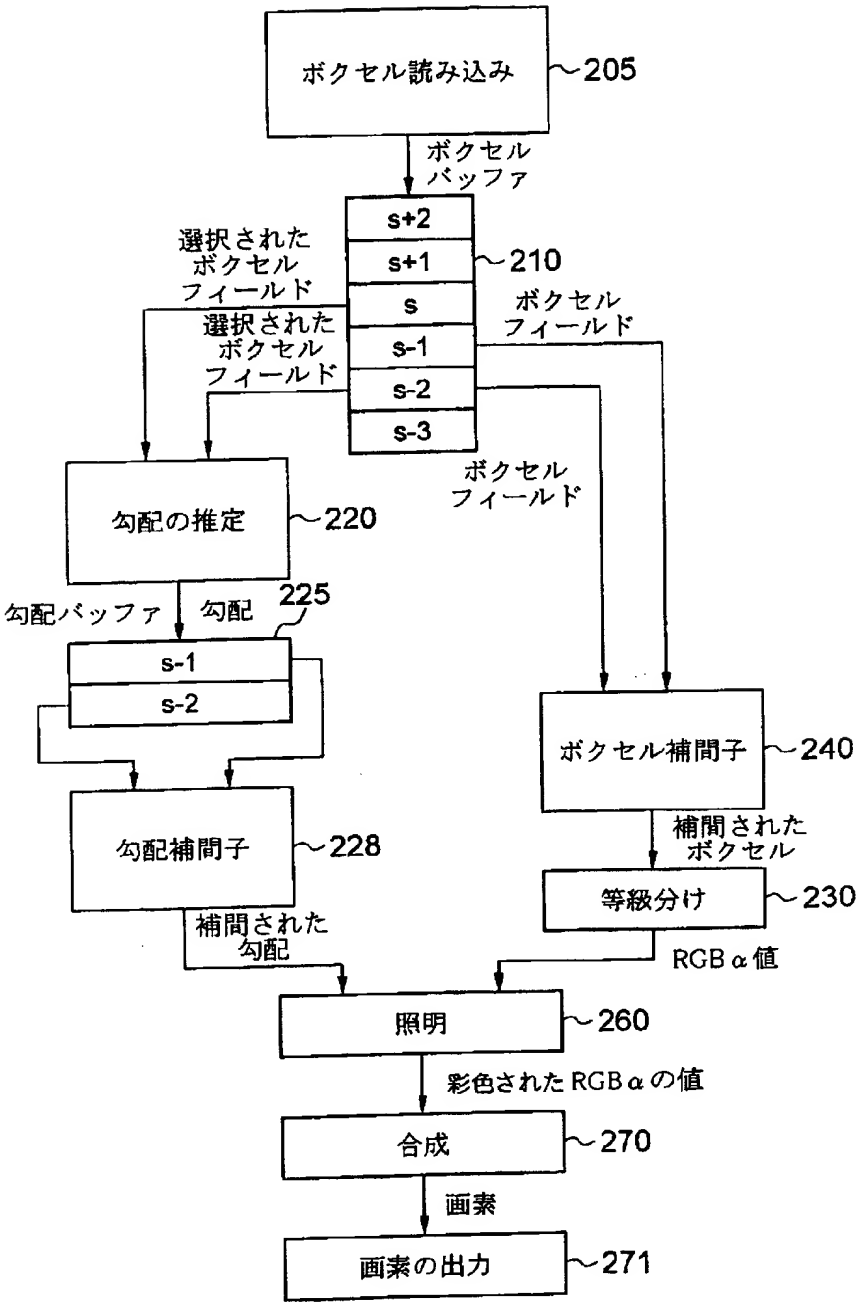
【図５】



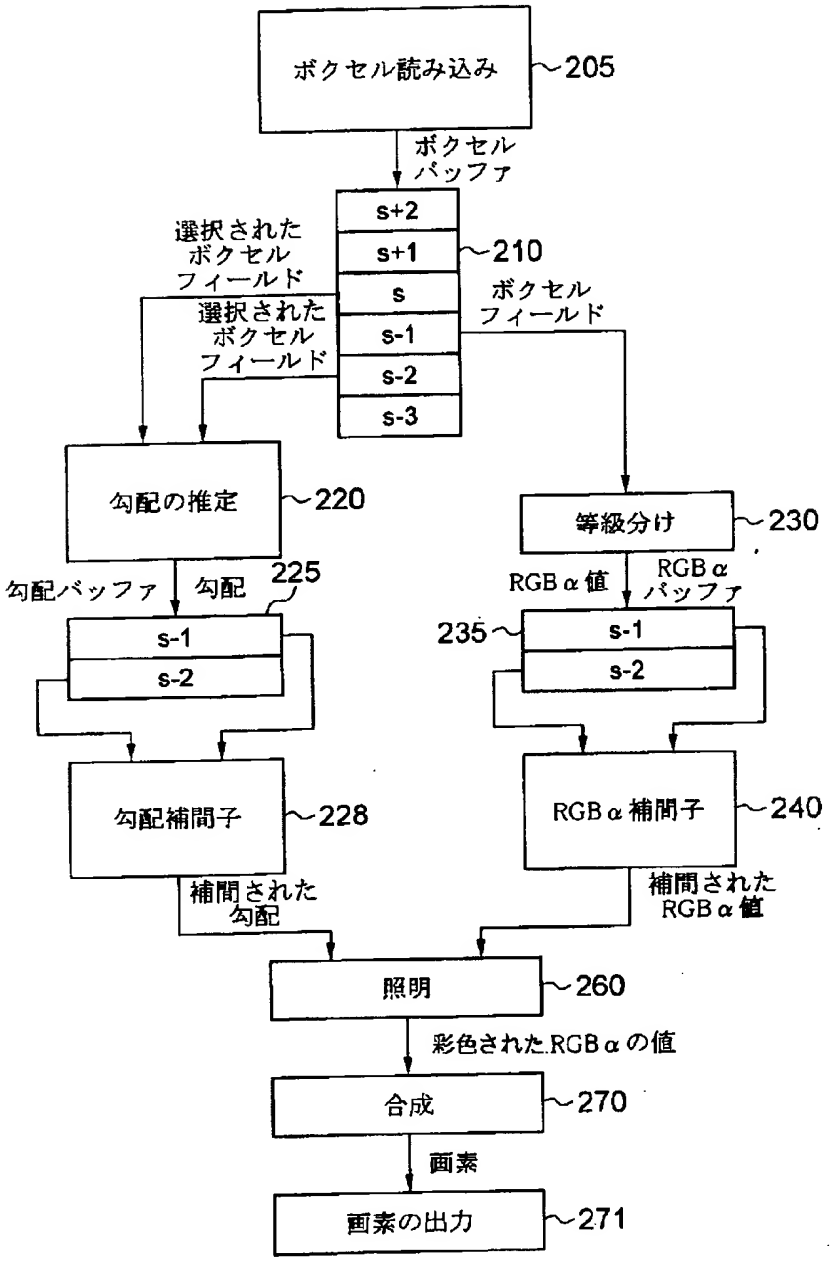
【図2】


201
メモリ
メモリ
ボクセルバッファ
210
200
xyz-勾配推定
220
ボクセルと勾配
221
GCI/GIC
マルチプレクサ
251
202
GCI/GIC
マルチプレクサ
252
202
等級分け
230
240
補間ユニット
(ボクセルとxyz-勾配)
GCI/GIC
マルチプレクサ
253
202
照明
260
彩色されたRGBαの値
合成
270
画素出力
271
画素メモリ
209

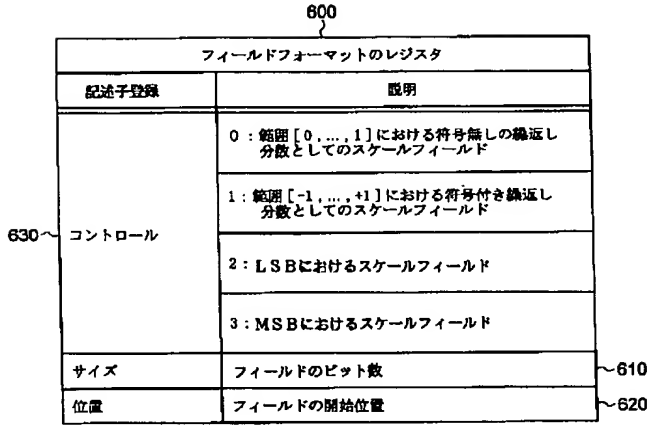
【図3】


ボクセル読み込み
205
ボクセル
バッファ
s+2
s+1
s
s-1
s-2
s-3
210
選択された
ボクセル
フィールド
選択された
ボクセル
フィールド
ボクセル
フィールド
ボクセル
フィールド
勾配の推定
220
勾配バッファ
勾配
225
s-1
s-2
勾配補間子
228
補間された
勾配
ボクセル補間子
240
補間された
ボクセル
等級分け
230
RGBα値
照明
260
彩色されたRGBαの値
合成
270
画素
画素の出力
271

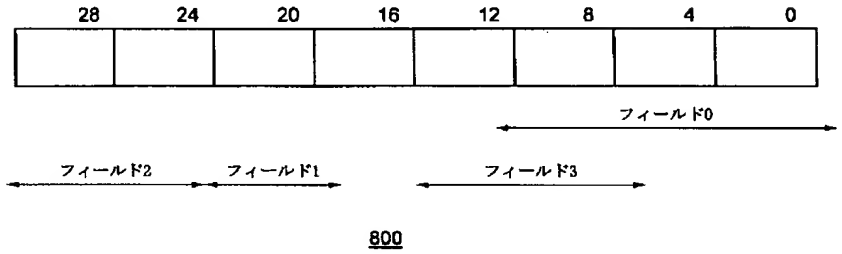
【図4】


ボクセル読み込み
205
ボクセル
バッファ
s+2
s+1
s
s-1
s-2
s-3
210
選択された
ボクセル
フィールド
選択された
ボクセル
フィールド
ボクセル
フィールド
勾配の推定
220
等級分け
230
勾配バッファ
勾配
225
RGBα値
RGBα
バッファ
235
s-1
s-2
s-1
s-2
勾配補間子
228
RGBα補間子
240
補間された
勾配
補間された
RGBα値
照明
260
彩色されたRGBαの値
合成
270
画素
画素の出力
271

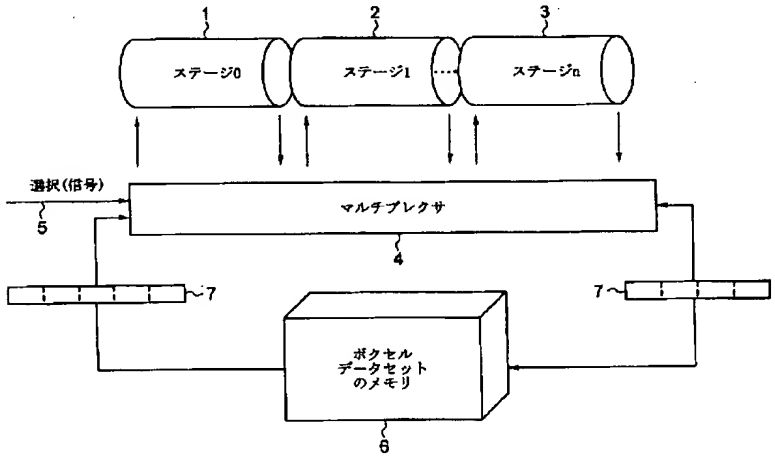
【図6】


600
フィールドフォーマットのレジスタ
記述子登録
説明
0：範囲[0,…,1]における符号無しの繰返し分数としてのスケールフィールド
1：範囲[-1,…,+1]における符号付き繰返し分数としてのスケールフィールド
2：LSBにおけるスケールフィールド
3：MSBにおけるスケールフィールド
630
コントロール
サイズ
フィールドのビット数
610
位置
フィールドの開始位置
620

【図8】


28
24
20
16
12
8
4
0
フィールド0
フィールド2
フィールド1
フィールド3
800

【図9】


1
2
3
ステージ0
ステージ1
ステージn
選択(信号)
5
マルチプレクサ
4
7
7
ピクセル
データセット
のメモリ
6

フロントページの続き

(71)出願人 501184364
2955 Campus Drive, Suite 325, San Mateo, California 94403, U.S.A.
(72)発明者 ラリー・ディー・シーラー
アメリカ合衆国、マサチューセッツ州、ボイルストン、リンデン・ストリート 198
(72)発明者 ジェームズ・エム・ニッテル
アメリカ合衆国、マサチューセッツ州、グラトン、ヒル・ロード 241
(72)発明者 ケニス・ダブリュ・コーラル
アメリカ合衆国、マサチューセッツ州、ランカスター、ルネンバーグ・ロード 2193
(72)発明者 ハリー・ガスパラキス
アメリカ合衆国、マサチューセッツ州、アクトン、デービス・ロード 9、ビー 7
(72)発明者 ヴィックラム・シンハ
アメリカ合衆国、マサチューセッツ州、レキシントン、カターディン・ドライブ 414
(72)発明者 ヴィシャル・シー・バティア
アメリカ合衆国、マサチューセッツ州、アーリントン、サマー・ストリート 478

Fターム(参考) 5B080 AA17 CA04 FA03 GA02

【外国語明細書】

1 Title of Invention

Volume Rendering Pipeline

2 Claims

We claim:

1. A volume rendering pipeline including a plurality of stages for rendering a volume data set, comprising:

a first multiplexer connecting an output of a first stage to an input of a second stage;

a second multiplexer connecting an output of the second stage to an input of the first stage;

a third multiplexer having inputs connected to the output of the first stage and the output of the second stage, the first, second, and third multiplexers responsive to a select signal for selecting an order of processing of the volume data set through the first and second stages.

2. The pipeline of claim 1 wherein the plurality of stages includes a gradient estimation stage, interpolation stage, classification stage, illumination stage, and compositing stage

3. The pipeline of claim 2 wherein the select signal connects the interpolation stage before the classification stage.

4. The pipeline of claim 2 wherein the select signal connects the classification stage before the interpolation stage.

5. The pipeline of claim 1 wherein a gradient interpolator using a gradient buffer operates in parallel with a RGBα interpolator using a RGBα buffer.

6. The pipeline of claim 1 wherein the first, second, and third multiplexers bypass a particular stage of the pipeline.

7. The pipeline of claim 1 further comprising:

a memory, connected to a first stage and last stage of the pipeline, the memory storing the volume data set before and after processing by the pipeline.

8. The pipeline of claim 1 wherein the memory stores a rendered image.

9. The pipeline of claim 1 wherein the volume data set includes a plurality of voxels, each voxel including plurality of fields.

10. The pipeline of claim 9 wherein each field has an associated offset and a width in the voxel.

11. The pipeline of claim 9 wherein a particular field stores a volume category.

12. The pipeline of claim 9 wherein each field is interpolated according to different associated interpolation function.

13. The pipeline of claim 2 wherein the gradient estimation stage extracts gradient components from a particular voxel having a plurality of fields.

3 Detailed Description of Invention

FIELD OF THE INVENTION

This invention relates generally to volume rendering, and more particularly, to a rendering pipeline wherein the order of processing in the pipeline stages can be user selectable.

BACKGROUND OF THE INVENTION

Introduction to Volume Rendering

Volume rendering is often used in computer graphics applications where three-dimensional data need to be visualized. The volume data can be scans of physical or medical objects, or atmospheric, geophysical, or other scientific models where visualization of the data facilitates an understanding of the underlying real-world structures represented by the data.

With volume rendering, the internal structure, as well as the external surface features of physical objects and models are visualized. Voxels are usually the fundamental data items used in volume rendering. A voxel is a data item that represents a particular three-dimensional portion of the object or model. The coordinates (x, y, z) of each voxel map the voxels to positions within the represented object or model.

A voxel represents some particular intensity value of the object or model. For a given prior art volume, intensity values can be a specific one of a number of different parameters, such as, density, tissue type,

elasticity, or velocity. During rendering, the voxel values are converted to color and opacity (RGBα) values, according to the intensity values, which can be projected onto a two-dimensional image plane for viewing.

One frequently used technique during rendering is ray-casting. A set of imaginary rays are cast through the array of voxels. The rays originate from a viewer's eye or from an image plane. The voxel values are re-sampled to points along the rays, and various techniques are known to convert the sampled values to pixel values. Alternatively, voxel values may be converted directly to RGBα voxels, which are then re-sampled along rays and accumulated to pixel values. In either case, processing of the volume data may proceed back-to-front, or front-to-back.

Rendering Pipeline

Volume rendering can be done by software or hardware. In one hardware implementation, the hardware is arranged as a multi-stage pipeline, see U.S. Patent Application 09/190,643 "Fast Storage and Retrieval of Intermediate Values in a Real-Time Volume Rendering System," filed by Kappler et al. on Nov. 12, 1998.

Figure 1 illustrates a pipeline 100 wherein voxel values are stored in a voxel memory 101. The voxel values are first read into a voxel buffer 110 of the pipeline as slices. The *z*-components of the gradients are estimated in stage 115 by taking central differences between voxels of different slices. Then, both the voxel values and the *z*-gradients are passed to an interpolation stage 120 that calculates these values at sample points along rays. Next, the *x*- and *y*-

components of the gradients are calculated from the interpolated sample values in stage 130. These, along with the sample values and the interpolated *z*-gradients are then passed to a classification stage 140, and then a shading stage 145, where an illumination process is applied to produce the RGBα values representing the illuminated samples. Finally, the illuminated samples are combined along rays in an compositing stage 150 to produce pixel values for the base plane stored in a pixel memory 109.

That pipeline structure suffers because the order of processing data is fixed by the arrangement of the various stages. Also, voxel values are interpolated so that only interpolated samples can be classified. It is not possible to concurrently render multiple volumes acquired from different scanning modalities. In addition, the format of the voxel data is fixed. Gradient fields are obtained from the fixed format voxel data. It is desired to improve on this prior art pipeline.

SUMMARY OF THE INVENTION

The invention provides a volume rendering pipeline including a plurality of processing stages. The stages can include a gradient estimation stage, an interpolation stage, a classification stage, an illumination stage, and a compositing stage. The stages are connected to each other by multiplexers.

A first multiplexer connects an output of a particular stage to an input of another stage, and a second multiplexer connects an input of a particular stage to an output of another stage. The multiplexers selectively connect the stages of the pipeline in a predetermined order to configure the rendering pipeline for processing a volume data set.

In one aspect of the invention, voxels of the volume data set are interpolated before classified, and in another aspect, the voxel are interpolated before they are classified.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Figure 2 shows a top-level block diagram of a configurable rendering pipeline 200 according to the invention. The pipeline 200 takes samples or voxels as input from a voxel memory 201, and stores pixels as output in a pixel memory 209. The voxels or samples are read into a voxel buffer 210 a volume slice at the time. Gradients are estimated for *xyz*-components in stage 220. Here, in contrast with the prior art, all gradient components are estimated on voxel or sample values, not on interpolated samples.

The voxels 221 are classified in stage 230. Interpolation of voxels and gradients takes place in stage 240. Gradient estimation and interpolation are linear operations and therefore can be interchanged. As an advantage of the invention, the application can determine the order in which the voxels and gradients are processed by the various stages. The reason for two different processing orders is a follows.

Scanners acquire sampled data by making physical measurements which typically represent an integral over a small region of space. Adjacent voxels often represent the integrals of overlapping regions. In these cases, it is more accurate to interpolate voxels first, then classify the result. In particular, a voxel at a boundary of, say, two different tissues is likely to represent an average of the physical properties of each tissue. When interpolating, all that is done is moving the points of average. To make this useful, the classification function should present a continuous transition from between the colors and opacities assigned to the two tissue types.

Sometimes, sampled data may be pre-segmented (manually or automatically) into distinct materials, e.g., bone, muscle, cartilage, etc. Given such data, it would be inappropriate to interpolate between two differently identified tissues to obtain a third tissue that is not really there. In these cases, it is better to classify the voxels first, then interpolate the resulting colors. This is not so much an average of the physical properties of adjacent tissue types, but rather a blend of the colors at their boundaries.

Therefore, the pipeline 200 includes a multiplexer 251 connecting the output from stage 240 to the input of stage 230. Similarly, a multiplexer 252 connects the output of stage 230 to the input of stage 240. Multiplexer 253 selects the output from either stage 230 or stage 240.

The order of connection of the stages depends on a selection signal (GCI/GIC) 202 to the three multiplexers 251-253. In either case, interpolated gradients and interpolated samples are passed to the illumination stage 260. Illuminated RGBα values are composited in stage 270, and output pixels 271 are stored in the pixel memory 209.

In the pipeline 200, the output of the gradient estimate stage 220 is the three components of gradients. These are passed to both the classification stage and the interpolation stage through the set of multiplexers. The output of each of these stages is each passed to the input of the other stage and also to the illumination stage 260 through another multiplexer.

By appropriately selecting the three multiplexers 251-253, it is possible to do classification before interpolation, or interpolation before classification, or only interpolation, or only classification. These different modes of operation is now described in greater detail.

GIC — Gradient, Interpolation, Classification

Figure 3 illustrates the rendering process for the GIC mode. This mode can be selected by signal 202. In the GIC mode, gradients are first estimated, then interpolation takes place on gradients and voxels in parallel, followed by classification of the voxels.

Rendering proceeds a section at a time, that is, in groups of 32×32 rays. In order to process a section of rays, slices of voxels are read into the voxel buffer 210 two slices at the time. Each slice has only the voxels needed for that section. The number of voxels per slice depends upon the view direction and resampling frequency. The maximum number of voxels needed is 37×37 voxels.

Reading voxels into the voxel buffer 210 can consumes the full bandwidth of the destination slices. Therefore, at any given time, one pair of slices is dedicated to receiving new voxels, while another two pairs of slices are dedicated to supplying the next stages of the pipeline, i.e., the gradient estimation 220 and the voxel interpolation stages 240.

In Figure 3, slices labeled *s +1* and *s +2* are receiving new voxels, while slices *s*, *s-1*, and *s-2* are providing voxels to the subsequent stages. Slice *s-3* was used during a previous slice and is currently empty. When the processing of slice *s-2* is complete, slice *s-2 is* also

marked as empty. Then, slices *s-2* and *s-3* will become the new destination for voxel reads, while the processing modules will start taking their input from slices *s +1*, *s*, and *s-1*.

The gradient estimation stage 220 estimates gradients at voxel points in slice *s-1* from voxel values in slices s and *s-2*. These gradients are then stored in a two slice gradient buffer 225 for interpolation by unit 228. Gradients for voxel points in slice *s-2* had been stored in the gradient buffer during a previous slice iteration.

Next, voxel interpolation 240 and gradient interpolation 228 proceed concurrently. That is, a slice of samples is obtained by interpolating voxel values from voxel slices *s -1* and *s -2*. Gradients at a sample points in the current slice of samples are obtained by interpolating the gradients of gradient slice *s -1* and gradient slice *s -2* in the gradient buffer. More than one slice of samples and interpolated gradients can be obtained from the same pair of voxel and gradient slices.

The interpolated voxel values are next applied to the classification stage 230, which converts the values to interpolated RGBα color values. The RGBα values and the gradients are next supplied to the illumination stage 260, and finally to the compositing stage 270.

In an optional mode, gradients are extracted directly from, for example, three voxel fields, instead of using the gradient estimation stage 220 to estimating gradients from any voxel field.

GCI - Gradient, Classification, Interpolation

Figure 4 illustrates the rendering algorithm for the GCI mode, also selected by signal 202. The voxel buffer 210 is filled as described above, and the gradient estimation process is also the same. The difference is in the voxel classification and interpolation side. In particular, raw voxels are taken from the voxel slice buffer, that is slice *s -1*, and immediately converted to RGBα values by the classification stage 230. These are then stored in a two slice RGBα buffer 235. Note, the RGBα buffer 235 is parallel to the two slice gradient buffer 225.

The two slices are then input to the RGBα interpolation stage 240, where interpolated RGBα values are produced in parallel with interpolating gradients. These are then applied with interpolated gradients to the illumination and compositing stages to produce the pixel output 271 of the base plane image. These last two modules are identical to those of Figure 2-3.

One additional optional mode is also possible. In this optional mode, multiple, for example, four, fields of the voxel are interpreted as raw RGBα values. In this case, the gradient may be estimated from the alpha (α) field.

It should be noted that for some volume data sets and rendering modes it may be best to estimate gradient in a later stage of the pipeline, for example, after classification.

It should be apparent that the multiplexers 251-253 can be replicated for other stages. Stages can selected and connected in a number of different orders to provide a reconfigurable rendering pipeline. Some stages can be de-selected to not process the volume data at all, for example, for some renderings gradient estimation and illumination may be skipped.

Flexible Voxel Format

As shown in Figure 5, a voxel 500 as used by the configurable rendering pipeline 200 includes a plurality of fields (V_1, ..., V_n) 501-209. Each of the fields 501-209 can be specified as an offset and width in the voxel 500. Any of the fields can overlap as shown for fields V_1 and V_2. The fields can be enumerated in any order.

The fields describe different attributes of a represented three-dimensional object or model. For example, if the object is a human head, then the fields 501-209 can respectively store intensity values acquired from CT, MRI, PET, SPECT, and ultrasound scans. i.e., each voxel may store five different scan intensity values in a single volume representation. Alternatively, the scans can be stored as multiple separate volumes wherein each voxel contributes one field.

Some fields can be category fields related to the way the volume is segmented, e.g., manual, semi-automatic, or automatic segmentation. In medical applications, segmentation can categorize bone, tissue, etc., In physical applications, segmentation can identify sub-assemblies or other parts to be rendered in a particular way. For physical models, the fields can store state variables used in scientific visualization, e.g., pressure, velocity, angular momentum, elasticity,

density, temperature, and viscosity. For any volume data set, the fields can also store RGBα values, depth, 3D stencil, shadows, fog, voxelized and embedded geometry volumes, or gradients.

For a multi-field voxel 500 according to our invention, the user can specify which fields to use for gradient calculations. For each component of the gradient, we can specify which of the voxel fields to use for that component.

For multi-field visualization, it is usually desirable to interpolate fields within voxels separately. Furthermore, each field within the voxel can have a different interpolation function applied, e.g., tri-linear for intensity fields, and nearest neighbor interpolation for category fields. The flexible voxels as described herein enable a common framework for treating all special cases of voxel formats in a uniform fashion.

Field Format Register

Figure 6 shows a field format register 600 that enables multiple field voxels according to the invention. In one embodiment, fields of voxels are defined by descriptors of the field format register. This is an 8-bit (7:0) descriptor defining the size of the field (in 4-bit nibbles) 610, the position of the field within its voxel 620, pixel (also in 4-bit nibbles), and what to do (control 630) when the field is a different size than specified for its intended use.

The control bits define how a field may adapted to fit the data path of the pipeline through which the field will pass. The field can be

changed by repeating fraction arithmetic, or by adding or removing bits from either the most significant or least significant end.

Control = 0: the field of the raw voxel is treated as an unsigned repeating fraction representing a number in the range [0...1]. To expand or shrink the repeating fraction to fit the data path, repeating fractional arithmetic is applied to scale and round, thereby representing the number with fewer or more bits of precession. Repeating fraction number representation is described in greater detail below.

Control = 1: the field of the voxel is treated as a signed repeating fraction in the range [-1...+1].

Control = 2: the field of the raw voxel is expanded or truncated in its least significant bits to fit the data path. The most significant bits are preserved.

Control = 3: the field of the raw voxel is expanded or truncated in its most significant bits to fit the data path. The least significant bits are preserved.

Repeating Fraction Number Representation

Many graphics applications use a fixed width binary number to represent color, transparency, or other parameters that have values in the range zero to one, inclusive.

Let R be the number of bits in the binary number and let V be the unsigned binary value stored in these bits. Then $F = V / (2^R - 1)$ is a

rational number in the range [*0..1*]. That is, when *V* equals zero, *F* equals zero, and when *V* equals its largest possible value, $(2^R - 1)$, *F* equals one. This representation is well known in the prior art. For example, the OpenGL Specification refers to it as a special kind of fixed point representation.

To clearly distinguish the representation described herein from ordinary fixed point representation, the term "repeating fractions" is used. The name term derives from the fact that expanding *F* into a fixed point binary fraction produces *0.VVVVVV...*, that is, a binary fraction that repeats the *R*-bit value *V* infinitely to the right of the binary point.

Repeating fractions can be represented with more than *R* bits and can even be signed. In that case, *R* is the "repeating precision" of the number, since *R* defines the implicit scale factor $(2^R - 1)$. This allows *F* to have values outside the range [*0...1*]. In this case, the binary fixed point representation consists of an integer value followed by an *R*-bit infinitely repeating binary value. Repeating fractions with the same precision may be added and subtracted in the same way as ordinary integers.

Other arithmetic operations, including changing the repeating precision, may be performed by first computing *F* for each repeating fraction, performing normal arithmetic, and then multiplying the resulting value by $(2^R - 1)$ for the repeating precision *R* of the result. More efficient forms exist for operations on repeating fractions. For example, doubling the repeating precision from *R* to *2R* simply requires computing $V + (V << R)$.

Voxel Formats

The pipeline according to the invention allows a wide range of input formats of voxels. Input voxels can be 8, 16, 32, or larger bit quantities. Fields can be 4, 8, 12, or 16 bits in width, and are aligned on a 4-bit boundary within the voxel. All voxels fields are scaled to fit their data paths, which are typically twelve bits wide.

The format of a voxel is described in a voxel format register 700. An example of the format of a 32-bit voxel 800 is illustrated in Figure 8. In Figure 7, the format register for the example voxel, *Field0* occupies bits 11:0, *Field1* occupies bits 23:20, *Field2* occupies bits 31:24, and *Field3* overlaps *Field0* and occupies bits 15:8. For the example shown in Figure 8, the descriptions of the fields in field format register are as follows:

Field3: Size = 1	Position = 2,
Field2: Size = 1	Position = 6,
Field1: Size = 0	Position = 5, and
Field0: Size = 2	Position = 0.

Advantages

Having flexible format voxels, and a reconfigurable pipeline enable a number of advantages. For example, one of the fields in a voxel can be used for category bits. These are extra bits in voxels that identify the voxel as part of some particular tissue, sub-assembly, or other partition of the volume. They are not interpolated, but they do contribute to the assignment of RGB α values to voxels. When category bits are used, classification usually precedes interpolation.

The flexible voxel format allows gradients to be estimated from any selected field of the voxel. In these cases, a convolution kernel, such as a central difference, is applied to the selected field of each voxel to obtain the x, y, and z-components of the gradient of that voxel. These gradients are then interpolated, in parallel with raw or classified voxels, in order to obtain gradients at sample points. These are then applied, along with RGBα values to the illumination stage 260, and finally to the compositing stage 270.

In some applications and for some volumes, it may be better to use gradients that are determined in some other manner, instead of estimating them from the fields of voxels on the fly. Flexible format voxels accommodate this mode. In particular, gradients may be precomputed for the voxels with a high level of precision. These precomputed gradients can be stored in one of the fields of the voxels. These precomputed gradients can bypass the estimation stage, but are interpolated and applied in the illumination stage 260 just as the gradients estimated on the fly.

In a further variation, precomputed gradient are applied to a convolution kernel for gradient estimation. This has the effect of taking the second partial derivative of the volume data set. By the same technique, higher order partial derivatives may be obtained. Such derivatives are useful for extracting weak surfaces in the volume data set. As described for Figure 9 below, this can be done by passing the volume data set through the pipeline (1-3) multiple times in a "multi-pass" operation. Each pass processing the volume data set with different set of rendering parameters.

The pipeline 200 processing flexible format voxels also admits volume data sets that are preclassified and presented as "RGBα" voxels. Two alternatives are useful. In one, gradient estimation and the classification and shading are be skipped entirely, and the voxels simply interpolated. In the other alternative, gradients are estimated from alpha values or a luminance function, and shading the RGBa values is done according to the illumination function. Note, the pipeline 200 according to the invention is also capable of interpolating both voxel-gradient values and RGBα values.

Multi-pass Volume Rendering

The rendering pipeline 200 can also write out a volume data set that has passed through some, but not necessarily all, of the stages of the pipeline 200 back to the voxel memory, see Figure 9. In an example use, the configured pipeline renders a set of voxels into RGBα values. Instead of composited along rays, the RGBα values are stored back to the memory 201 as RGBα voxels in a three-dimensional voxel array. Then, the volume is rerendered with a different set of rendering parameters. This process can repeat until a final volume is present in memory 201. The final accumulated values are then rendered one final time to generate an image in the pixel memory 209.

This technique enables a number of features, such as fast resampling. With fast resampling, the volume data set can be resampled to a different resolution using the speed and power of the volume rendering pipeline and volume memory, instead of relying upon the host processor and software.

Multi-pass rendering can be used to produce complex shadows on the volume. One pass is needed for each light source. Each pass is accumulated in the output volume data set, and then a final pass interpolates the results and projects them onto the base plane. In multi-pass rendering, the volume data set output from a current pass is combined with an already existing volume data set in the memory. This sometimes requires a read-modify-write operation rather than a simply write operation.

Multi-Channel Rendering

The pipeline 200 can also process multi-channel data sets. Some scanning techniques, such as, ultrasound and seismic applications, have data of
more than one type, each with its own classification. During rendering these data sets are be superimposed on each other, and combined, voxel-by-voxel.

More particularly, the compositing stage 270 is able to operate in either of two modes. In a first mode, the RGBα values are combined with previously stored pixel values, and in a second mode, a ray of RGBα values is accumulated, and the result is combined with a previously stored pixel value from some previous pass.

Pipeline General Structure

Generally, the relation of the pipeline 200 and the memory 201 can be as shown in Figure 9. Here, the stages ($stage_0$, $stage_1$, ..., $stage_n$) 1-3 of the pipeline 200 are connected to each other by multiplexers 4 so that for a particular rendering application, the stages are ordered by a

select signal 5. The input to the pipeline is a raw volume data set stored in a memory 6. The output of the pipeline is a modified volume data set. Individual data items of the volume data set passed between the pipeline and the memory are flexible format voxels 7. The volume data set can be processed by multiple passes.

It is to be understood that various other adaptations and modifications may be made within the spirit and scope of the invention. Therefore, it is the object of the appended claims to cover all such variations and modifications as come within the true spirit and scope of the invention.

4 Brief Descripton of Drawings

Figure 1 is a block diagram of a prior art rendering pipeline;
Figure 2 is a block diagram of a configurable rendering pipeline;
Figure 3 is a pipeline with interpolation before classification;
Figure 4 is a pipeline with classification before interpolation;
Figure 5 is a block diagram of a flexible format voxel.
Figure 6 is a block diagram of a field format register;
Figure 7 is a block diagram of a voxel format register;
Figure 8 is an example formatted voxel;
Figure 9 is a block diagram of a configurable rendering pipeline connected to a volume memory.

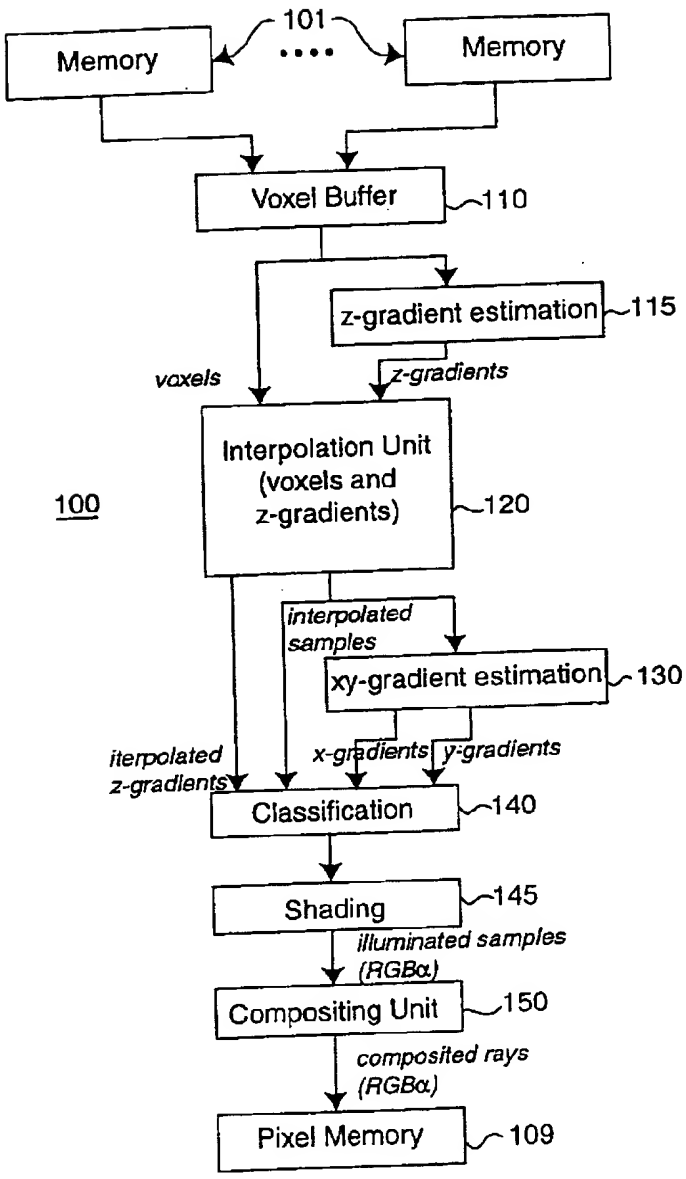

101
Memory
Memory
Voxel Buffer
110
z-gradient estimation
115
voxels
z-gradients
Interpolation Unit
(voxels and
z-gradients)
120
100
interpolated
samples
xy-gradient estimation
130
iterpolated
z-gradients
x-gradients
y-gradients
Classification
140
Shading
145
illuminated samples
(RGBα)
Compositing Unit
150
composited rays
(RGBα)
Pixel Memory
109

FIG. 1

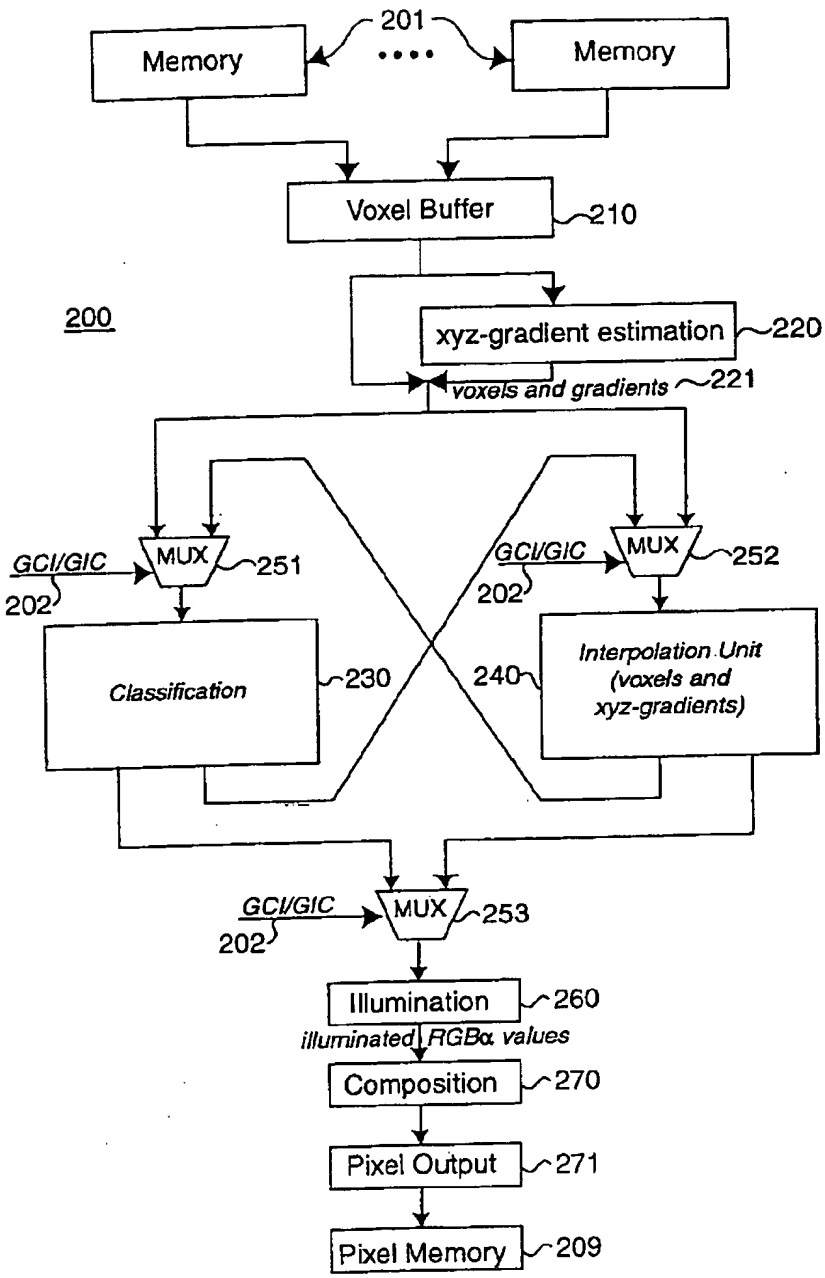

201
Memory
Memory
Voxel Buffer
210
200
xyz-gradient estimation
220
voxels and gradients
221
GCI/GIC
202
MUX
251
GCI/GIC
202
MUX
252
Classification
230
240
Interpolation Unit
(voxels and
xyz-gradients)
GCI/GIC
202
MUX
253
Illumination
260
illuminated RGBα values
Composition
270
Pixel Output
271
Pixel Memory
209

FIG. 2

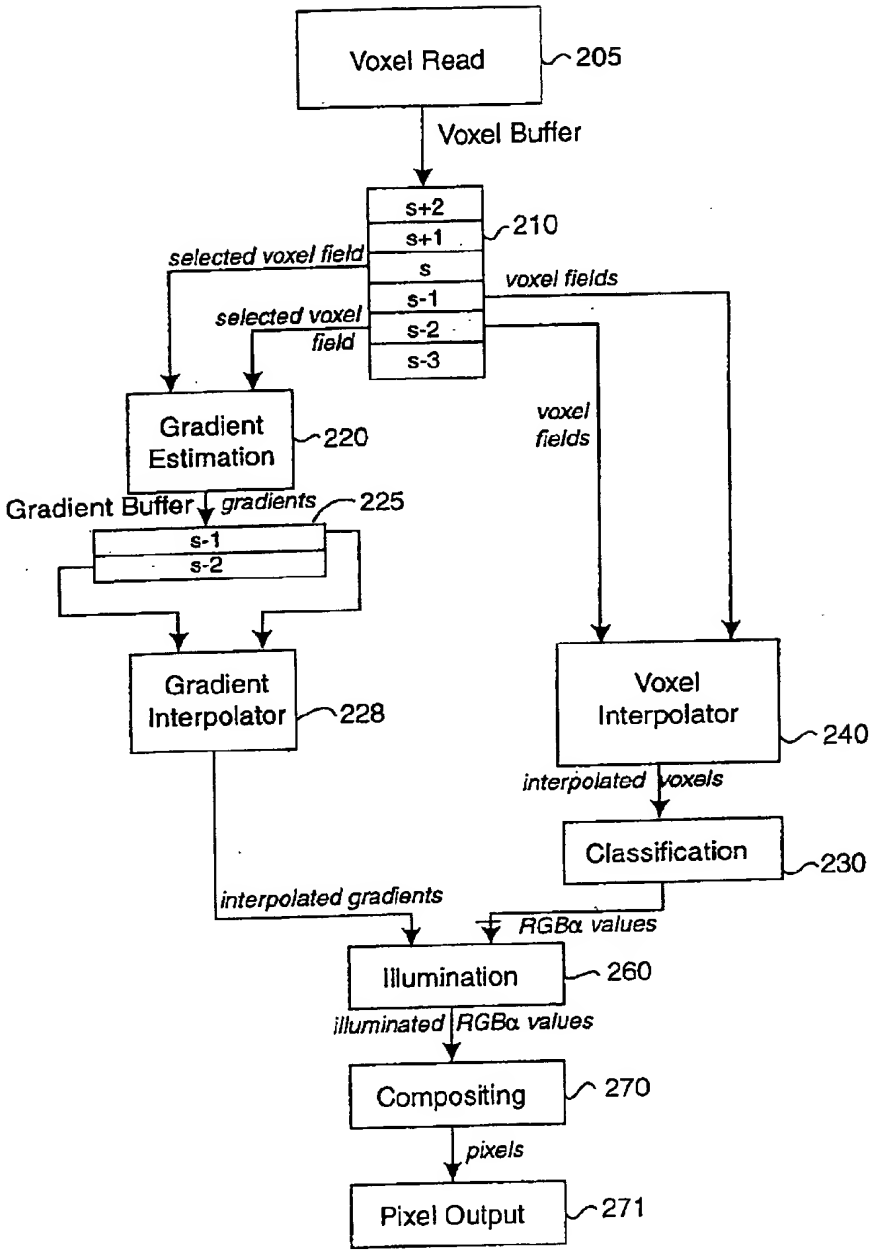

Voxel Read
205
Voxel Buffer
s+2
s+1
s
s-1
s-2
s-3
210
selected voxel field
voxel fields
selected voxel field
Gradient Estimation
220
voxel fields
Gradient Buffer
gradients
225
s-1
s-2
Gradient Interpolator
228
Voxel Interpolator
240
interpolated voxels
Classification
230
interpolated gradients
RGBα values
Illumination
260
illuminated RGBα values
Compositing
270
pixels
Pixel Output
271

FIG. 3

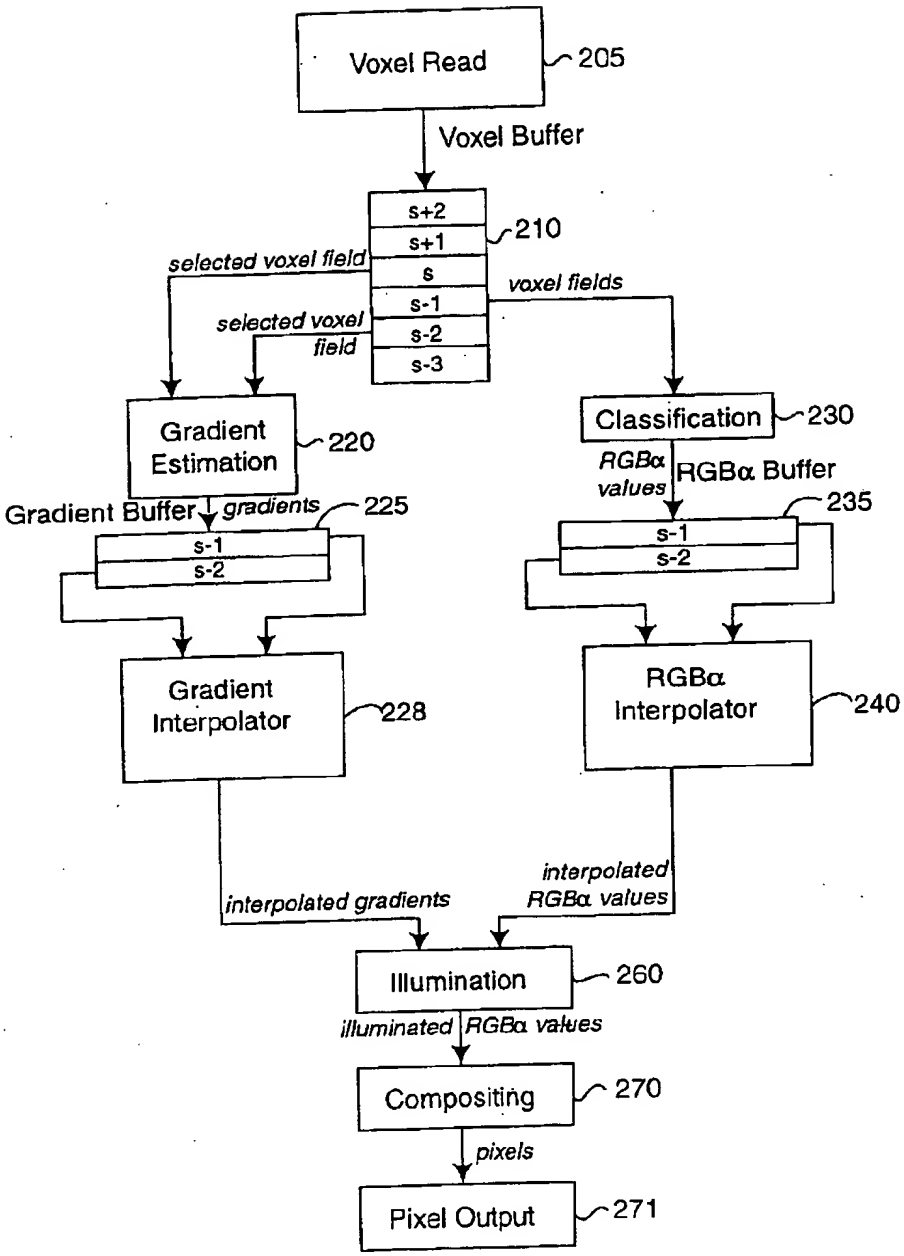

Voxel Read
205
Voxel Buffer
s+2
s+1
s
s-1
s-2
s-3
210
selected voxel field
selected voxel field
voxel fields
Gradient Estimation
220
Classification
230
RGBα values
RGBα Buffer
Gradient Buffer
gradients
225
235
s-1
s-2
s-1
s-2
Gradient Interpolator
228
RGBα Interpolator
240
interpolated gradients
interpolated RGBα values
Illumination
260
illuminated RGBα values
Compositing
270
pixels
Pixel Output
271

FIG. 4

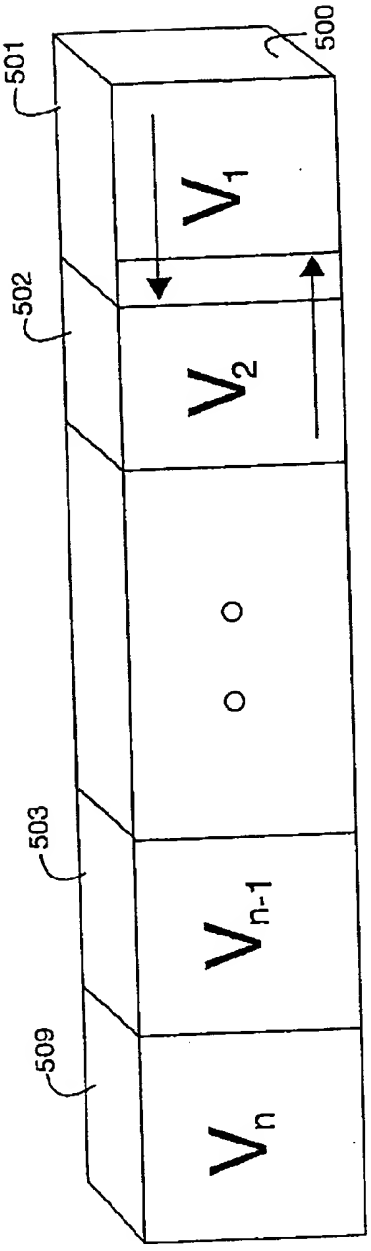


FIG. 5

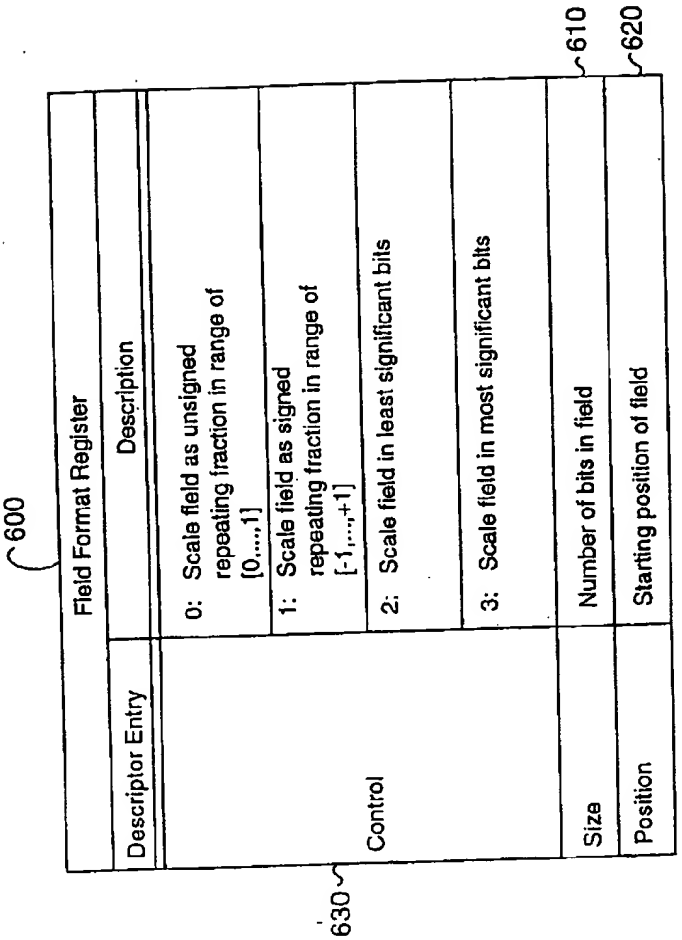

600
Field Format Register
Descriptor Entry
Description
0: Scale field as unsigned repeating fraction in range of [0,...,1]
1: Scale field as signed repeating fraction in range of [-1,...,+1]
2: Scale field in least significant bits
3: Scale field in most significant bits
Control
630
Size
Number of bits in field
610
Position
Starting position of field
620

FIG. 6

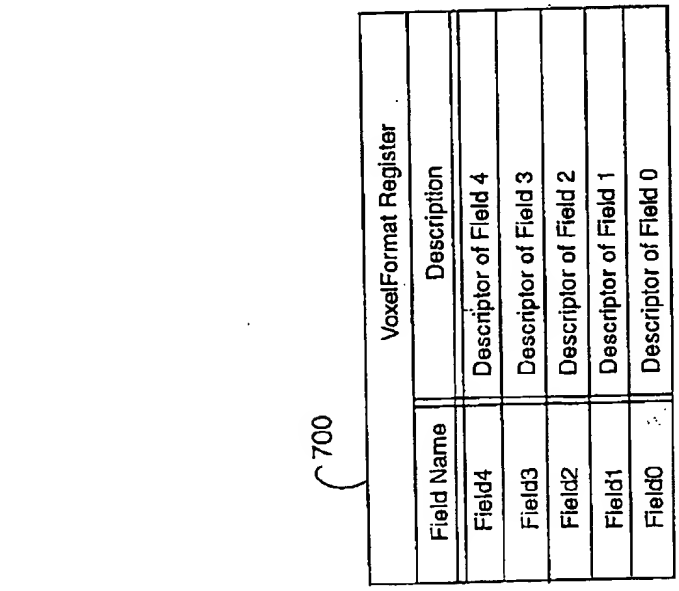

700
VoxelFormat Register
Field Name
Description
Field4
Descriptor of Field 4
Field3
Descriptor of Field 3
Field2
Descriptor of Field 2
Field1
Descriptor of Field 1
Field0
Descriptor of Field 0

FIG. 7

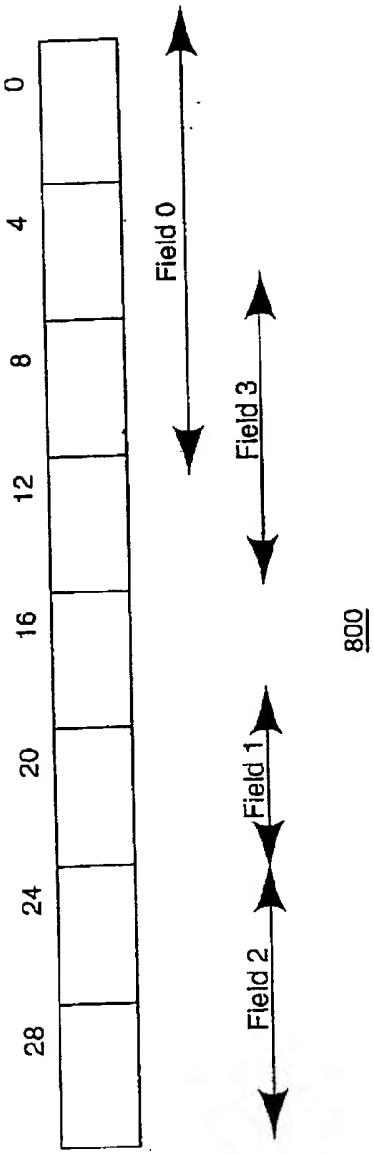


FIG. 8

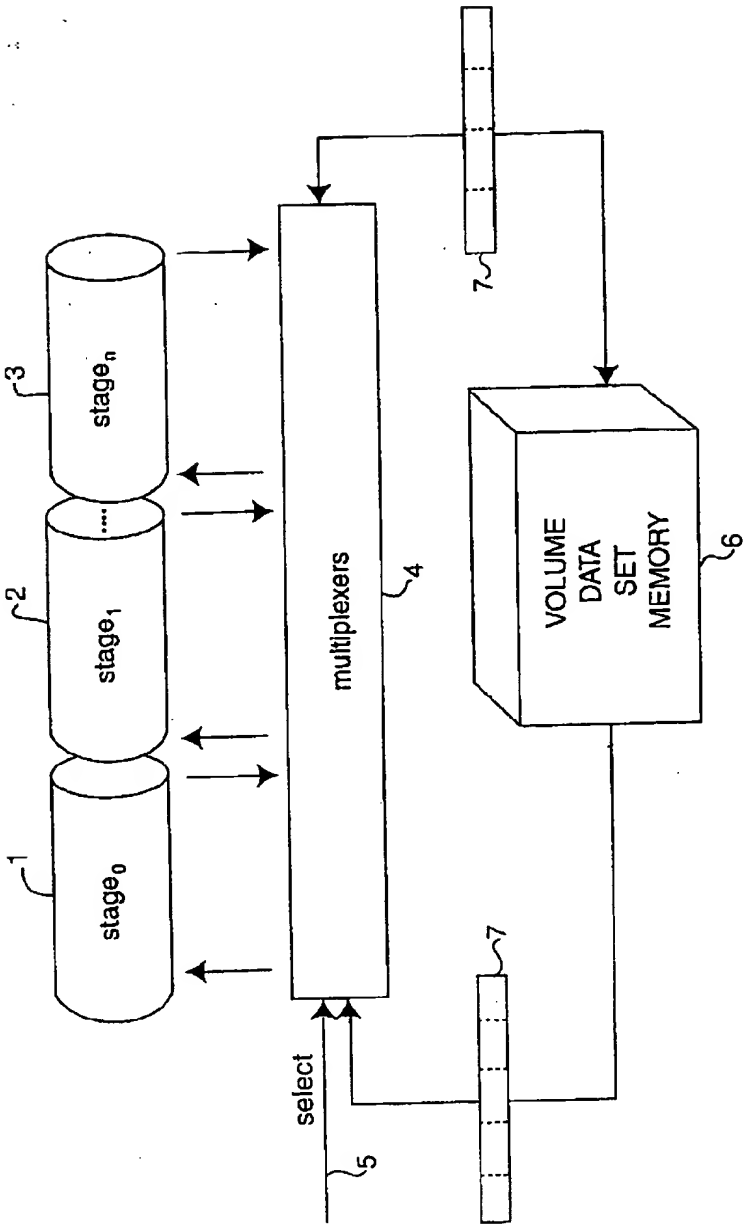

stage$_n$
stage$_1$
stage$_0$
3
2
1
multiplexers
4
VOLUME DATA SET MEMORY
6
7
7
select
5

FIG. 9

1 Abstract

A volume rendering pipeline includes a plurality of processing stages such as a gradient estimation stage, an interpolation stage, a classification stage, an illumination stage, and a compositing stage. The stages are connected to each other by multiplexers. A first multiplexer connects an output of a first stage to an input of a second stage. A second multiplexer connects an output of the second stage to an input of the first stage. A third multiplexer has inputs connected to the output of the first stage and the output of the second stage, the first, second, and third multiplexers are responsive to a select signal to configure the stages of the rendering pipeline for processing the volume data set.

2 Representative Drawing

FIG.1